Freeform Search

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: L19 and verif$

Display: 50 **Documents in Display Format:** FRO **Starting with Number** 1

Generate: Hit List Hit Count Side by Side Image

Search Clear Interrupt

Search History

DATE: Tuesday, September 12, 2006 Purge Queries Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L20	L19 and verif$	2	L20
L19	L18 and match$	2	L19
L18	L16 and (formats)	11	L18
L17	L16 and (specific near format)	0	L17
L16	L15 and (intelligent near design)	11	L16
L15	L14 and automat$	15	L15
L14	L13 and (library near format)	16	L14
L13	L12 and (electronic near design)	785	L13
L12	(computer near design$)	15675	L12
L11	L10 and query	5	L11
L10	l6 and (data near model)	5	L10
L9	L8 and (data near model)	4	L9
L8	L1 and (intelligent near design)	19	L8
L7	L6 and (intelligent near design)	0	L7
L6	L1 and (import near application)	37	L6
L5	L2 and (import near application)	0	L5

L4	L3 and (import near application)	0	L4
L3	L2 and (compiler or verifier)	8	L3
L2	L1 and (format near reader)	26	L2
L1	software near application	27934	L1

END OF SEARCH HISTORY

Freeform Search

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: L29 and ((multiple or plurality) near formats)

Display: 50 **Documents in Display Format:** FRO **Starting with Number** 1

Generate: Hit List Hit Count Side by Side Image

Search Clear Interrupt

Search History

DATE: Tuesday, September 12, 2006 Purge Queries Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=OR		
L31	L29 and ((multiple or plurality) near formats)	3	L31
L30	L29 and (different near (applications or softwares))	6	L30
L29	l13 and collaborat$	33	L29
L28	L13 and (collaborative near work)	0	L28
L27	L25 and (groupware)	1	L27
L26	L25 and (single near software)	0	L26
L25	l13 and (group near users)	31	L25
L24	L23 and (different near applications)	1	L24
L23	L22 and shar$	14	L23
L22	L14 and (group$)	15	L22
L21	L14 and (cooperat$ near work)	0	L21
L20	L19 and verif$	2	L20
L19	L18 and match$	2	L19
L18	L16 and (formats)	11	L18
L17	L16 and (specific near format)	0	L17
L16	L15 and (intelligent near design)	11	L16

L15	L14 and automat$	15	L15
L14	L13 and (library near format)	16	L14
L13	L12 and (electronic near design)	785	L13
L12	(computer near design$)	15675	L12
L11	L10 and query	5	L11
L10	l6 and (data near model)	5	L10
L9	L8 and (data near model)	4	L9
L8	L1 and (intelligent near design)	19	L8
L7	L6 and (intelligent near design)	0	L7
L6	L1 and (import near application)	37	L6
L5	L2 and (import near application)	0	L5
L4	L3 and (import near application)	0	L4
L3	L2 and (compiler or verifier)	8	L3
L2	L1 and (format near reader)	26	L2
L1	software near application	27934	L1

END OF SEARCH HISTORY

L31: Entry 3 of 3 File: USPT Dec 12, 2000

DOCUMENT-IDENTIFIER: US 6161211 A
TITLE: Method and apparatus for automated circuit design

Parent Case Text (4):
This invention is also related to U.S. patent application Ser. No. 08/958,436, filed on the same day as this patent application, naming J. Tse et al. as inventors, and entitled "FITTING FOR INCREMENTAL COMPILATION OF ELECTRONIC DESIGNS." That application is incorporated herein by reference in its entirety and for all purposes.

Parent Case Text (5):
This invention is also related to U.S. patent application Ser. No. 08/958,670, filed on the same day as this patent application, naming D. Mendel as inventor, and entitled "PARALLEL PROCESSING FOR COMPUTER ASSISTED DESIGN OF ELECTRONIC DEVICES." That application is incorporated herein by reference in its entirety and for all purposes.

Parent Case Text (6):
This invention is also related U.S. patent application Ser. No. 08/958,626, filed on the same day as this patent application, naming F. Heile et al. as inventors, and entitled "INTERFACE FOR COMPILING DESIGN VARIATIONS IN ELECTRONIC DESIGN ENVIRONMENTS." That application is incorporated herein by reference in its entirety and for all purposes.

Parent Case Text (12):
This invention is also related to U.S. patent application Ser. No. 08/957,957 now U.S. Pat. No. 5,983,277, filed on the same day as this patent application, naming F. Heile et al. as inventors, and entitled "WORKGROUP COMPUTING FOR ELECTRONIC DESIGN AUTOMATION." That application is incorporated herein by reference in its entirety and for all purposes.

Parent Case Text (15):
This invention is also related to U.S. patent application Ser. No. 08/958,431, filed on the same day as this patent application, naming F. Heile as inventor, and entitled "ELECTRONIC DESIGN AUTOMATION TOOL FOR DISPLAY OF DESIGN PROFILE." That application is incorporated herein by reference in its entirety.

Brief Summary Text (3):
There are a number of commercially available software design tools for designing programmable logic devices. The level of expertise required to use these tools and the complexity of the tools themselves vary rather dramatically. Some tools employ a syntax-heavy approach which requires sophisticated programming skills and intimate knowledge of a text editor. Others employ a more userfriendly visual approach which relies on the manipulation of visual objects using a graphic editor. In the past, the common denominator for such design tools has been that each is typically employed to specify the circuit level implementation of the logic device. The higher levels of the design process were left to the discretion and idiosyncrasies of the individual designer who would, in many cases, implement and document the high level design using paper and pencil. Not only can this result in inconsistent documentation of the design, it makes it difficult for design information to be shared among collaborating designers.

Brief Summary Text (8):
According to the present invention a design methodology is provided by which the design of a

L31: Entry 2 of 3 File: USPT Oct 14, 2003

DOCUMENT-IDENTIFIER: US 6634008 B1
TITLE: Methodology server based integrated circuit design

Abstract Text (1):
An environment for designing integrated circuits. Computers include browsers for displaying pages of forms, with the computers in communication with a methodology server and a compute server. The methodology server contains design methodologies accessed by the computers, with the design methodologies defining steps of designing and testing of integrated circuits. The computers or methodology server are also in communication with a compute server. The compute server executes electronic design automation tools as requested.

Brief Summary Text (12):
The present invention therefore provides an environment for designing integrated circuits. In one embodiment the present invention comprises an integrated design environment for the design and test of integrated circuits, the integrated circuits being comprised of blocks. The integrated design environment includes a plurality of computers, with the computers including a browser for the display of pages including forms. The integrated design environment also includes at least one methodology server in communication with the plurality of computers. The methodology server includes a page generator generating the pages including forms and additionally including programs responsive to submission of information from the computers using the pages including forms. In one embodiment the programs responsive to submission of information comprise common gateway interface (CGI) programs or scripts. The integrated design environment also includes at least one compute server in communication with the methodology server. The compute serve includes an electronic automation tool, and the compute server executes the electronic design automation tool in response to a request generated by a program resident on the methodology server.

Detailed Description Text (19):
The interface and flow control tool encompasses HTML pages and CGI scripts. The HTML pages include input forms for defining methodologies, including steps of methodologies, as well as chip and block home pages and executable methodologies. The CGI scripts receive and act on data input to the input forms to create files defining methodologies, chips and blocks, and executable methodologies attached to chips and blocks. The CGI scripts also cause execution of electronic design automation (EDA) tools residing on the compute servers (illustrated in FIG. 2).

Detailed Description Text (24):
In an alternative embodiment, the design tool is executed in real time, with the execution of the design tool being displayed to a designer at the workstation using an X window. FIG. 6 illustrates four communication methods used by alternative embodiments of the present invention. Each of the four embodiments include an electronic design automation (EDA) tool 2301 in communication with an application 2309. The application may be, for example, another EDA tool. The EDA tool may, for example, be a Verilog reader which reads a Verilog file and provides the information contained in the Verilog file to an application. The application may, for example, utilize the information from the Verilog file to compile the design into representative hardware elements.

Detailed Description Text (91):
The process in step 527 may convert the modified design methodology in XML format to the design methodology format where the design methodology comprises multiple files. In an embodiment of

L20: Entry 1 of 2 File: USPT Apr 10, 2001

DOCUMENT-IDENTIFIER: US 6216252 B1
TITLE: Method and system for creating, validating, and scaling structural description of electronic device

Abstract Text (2):
Techniques for scaling of a model design to provide a scaled design are described whereby parameters of a model design such as size, circuit complexity, interconnection density, number of I/O connections, etc., can be scaled to produce a scaled version of the design. The scaling techniques employ multi-level hierarchical module replication to produce fully-functional scaled designs which closely match the function of the model design. Test vectors for the scaled designs can be readily obtained by altering test vectors for the model design to account for the replicated modules.

Brief Summary Text (5):
One of the most difficult problems in design automation is the inability to get timing closure at even the gate level effectively. This forces designers to do two designs: logic design and timing design. Otherwise, the designer simply over-designs the circuits, because the best case timing is much different from the worst case timing. In other cases, designers insist on control of device layout so that they can evaluate all of the tradeoffs between implementation and timing.

Brief Summary Text (6):
Present computer aided design (CAD) systems for the design of electronic circuits, referred to as ECAD or Electronic CAD systems, assist in the design of electronic circuits by providing a user with a set of software tools running on a digital computer with a graphical display device. Typically, five major software program functions run on the ECAD system: a schematic editor, a logic compiler, a logic simulator, a logic verifier, and a layout program. The schematic editor program allows the user of the system to enter and/or modify a schematic diagram using the display screen, generating a net list (summary of connections between components) in the process. The logic compiler takes the net list as an input, and using a component database puts all of the information necessary for layout, verification and simulation into a schematic object file or files whose format(s) is(are) optimized specifically for those functions. The logic verifier checks the schematic for design errors, such as multiple outputs connected together, overloaded signal paths, etc., and generates error indications if any such design problems exist. The logic simulator takes the schematic object file(s) and simulation models, and generates a set of simulation results, acting on instructions initial conditions and input signal values provided to it either in the form of a file or user input. The layout program generates data from which a semiconductor chip (or a circuit board) may be laid out and produced.

Brief Summary Text (13):
Generally, there is a need to make improvements in the tools and techniques used to implement electronic designs (i.e., the designs of circuits and systems), in other words to assist the designer in the task of creating an efficient physical implementation (e.g., in silicon) of a concept (e.g., design specification). Some specific ones of these needs are discussed in greater detail hereinbelow.

Brief Summary Text (16):
It is a further object of the present invention to provide a technique for automatically translating behavioral descriptions of a circuit or system into physical implementations

L20: Entry 2 of 2 File: USPT Jan 28, 1997

DOCUMENT-IDENTIFIER: US 5598344 A
TITLE: Method and system for creating, validating, and scaling structural description of electronic device

Abstract Text (1):
A methodology for generating structural descriptions of complex digital devices from high-level descriptions and specifications. The methodology uses a systematic technique to map and enforce consistency of the semantics imbedded in the intent of the original, high-level descriptions. The design activity is essentially a series of transformations operating upon various levels of design representations. At each level, the intended meaning (semantics) and formal software manipulations are captured to derive a more detailed level describing hardware meeting the design goals. Important features of the methodology are: capturing the users concepts, intent, specification, descriptions, constraints and trade-offs; architectural partitioning; what-if analysis at a high level; sizing estimation; timing estimation; architectural trade-off; conceptual design with implementation estimation; and timing closure. The methodology includes using estimators, based on data gathered over a number of realized designs, for partitioning and evaluating a design prior to logic synthesis. From the structural description, a physical implementation of the device is readily realized. Techniques for scaling of a model design to provide a scaled design are provided whereby parameters of a model design such as size, circuit complexity, interconnection density, number of I/O connections, etc., can be scaled to produce a scaled version of the design. The scaling techniques employ multi-level hierarchical module replication to produce fully-functional scaled designs which closely match the function of the model design. Test vectors for the scaled designs can be readily obtained by altering test vectors for the model design to account for the replicated modules.

Brief Summary Text (5):
One of the most difficult problems in design automation is the inability to get timing closure at even the gate level effectively. This forces designers to do two designs: logic design and timing design. Otherwise, the designer simply over-designs the circuits, because the best case timing is much different from the worst case timing. In other cases, designers insist on control of device layout so that they can evaluate all of the tradeoffs between implementation and timing.

Brief Summary Text (6):
Present computer aided design (CAD) systems for the design of electronic circuits, referred to as ECAD or Electronic CAD systems, assist in the design of electronic circuits by providing a user with a set of software tools running on a digital computer with a graphical display device. Typically, five major software program functions run on the ECAD system: a schematic editor, a logic compiler, a logic simulator, a logic verifier, and a layout program. The schematic editor program allows the user of the system to enter and/or modify a schematic diagram using the display screen, generating a net list (summary of connections between components) in the process. The logic compiler takes the net list as an input, and using a component database puts all of the information necessary for layout, verification and simulation into a schematic object file or files whose format(s) is(are) optimized specifically for those functions. The logic verifier checks the schematic for design errors, such as multiple outputs connected together, overloaded signal paths, etc., and generates error indications if any such design problems exist. The logic simulator takes the schematic object file(s) and simulation models, and generates a set of simulation results, acting on

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#
End of Result Set

Generate Collection Print

L27: Entry 1 of 1 File: USPT Dec 7, 1999

DOCUMENT-IDENTIFIER: US 5999911 A
TITLE: Method and system for managing workflow

Brief Summary Text (5):
The electronic design automation (EDA) field has produced a number of computer-based design tools to assist in the development of complex electronics. To list a few examples, there are computer-based tools to synthesize, simulate, and test an electronic circuit design. These tools have enabled design teams to decrease significantly the design cycle for complex electrical components.

Brief Summary Text (6):
Even with the array of design tools available to assist the design team, the design problem is still very difficult to manage. Many of today's computerbased design tools only assist in a single step in the design process such as helping design the physical layout of components on a circuit board or simulating the behavior of a circuit design. In most electronic design problems however, there are a number of steps, each potentially involving a number of designers and a large amount of data. In this complex design environment, it is very difficult to track the design steps and to organize the design data generated from those steps. Thus, while today's design tools help developers and engineers perform their design tasks, they do not adequately address the issue of managing the design process as a whole.

Brief Summary Text (7):
Groupware presents an alternative solution for helping designers manage shared data relating to a design problem. Groupware products facilitate the sharing of information in the form of documents among members of an organization. They enable users of computers in a network to edit and view information stored in shared documents.

Brief Summary Text (8):
Though effective at maintaining shared information, current groupware tools do not guide designers through related process steps. Groupware does not allow a designer to define a model for an arbitrary process. More specifically, groupware does not allow a user to describe steps in the process and the dependencies among these steps. In a typical design process, there are a number of steps which must be performed in a particular order. As designers perform these steps, they create or transform design data. Groupware cannot effectively model a design process because it cannot model the dependency among steps, and it cannot manage the creation and transformation of data from process steps.

Brief Summary Text (9):
Apart from groupware, a number of computerbased tools in the EDA marketplace purportedly address the concept of a design process. These design tools fall into one of the following categories: (1) Tool Launchers; (2) Tool Sequencers; and (3) Flow Sequencers.

Detailed Description Text (14):
It should be understood that FIG. 1 is a block diagram illustrating the basic elements of a computer system; the figure is not intended to illustrate a specific architecture for a computer system 20. For example, no particular bus structure is shown because various bus structures known in the field of computer design may be used to interconnect the elements of the computer system in a number of ways, as desired. CPU 28 may be comprised of a discrete ALU

L11: Entry 3 of 5 File: USPT Nov 8, 2005

DOCUMENT-IDENTIFIER: US 6963930 B2
TITLE: Automatic transfer and expansion of application-specific data for display at a website

Brief Summary Text (6):
In addition, there are many applications for many different types of data. For example, mechanical engineers on a design teams may use three-dimensional CAD systems to develop floor plans that position components of the system on a shop floor or in another appropriate environment to provide easy access and optimize use of available space. Process engineers use process modeling software tools to generate a computerized representation of the system environment to design processes that use raw materials efficiently and eliminate significant bottlenecks. Electrical engineers use software applications to develop electrical plans that ensure, among other things, that the articles in the environment have access to appropriate electrical inputs. Software development teams use project task applications to specify task interrelationships and start and end dates. Technical writers use as technical documentation software to access and modify technical documents having chapters, sections, and subsections, and associated tables-of contents, which must be updated and modified as the document is revised.

Brief Summary Text (15):
Accordingly, a need exists for a software application that can publish hierarchical data that has been obtained from any of a number of applications and have the ability to update that data.

Drawing Description Text (6):
FIG. 4 is an example of data produced by the Microsoft Project software application.

Drawing Description Text (9):
FIG. 7 is a user-interface display on which the user indicates that all of the configuration options have been set, and the user is ready to complete the data model construction and web site generation.

Detailed Description Text (6):
As shown at 310 in FIG. 2A, after producing application-specific structured data at a client, the user executes a browser program at the server and navigates in the web browser to the system web page from the server that has an "Inbox Control" icon. The "Inbox Control" is an ActiveX control that functions as a "drop" target in a Graphical User Interface image presented by the website to the user at the client. The user "drags and drops" the application-specific data into the Inbox Control 110 of the website. Although in the preferred embodiment, data is imported into the website by dragging and dropping a file representation into the Inbox Control, other methods to import application-specific data onto a website are also possible, such as conventional browsing in a window.

Detailed Description Text (8):
When an application-specific data file is dragged and dropped into the "Inbox", an appropriate client component, capable of building an XML file for a specific dropped file type, is selected. The resulting XML file is packaged at the client and transferred to the server for processing as a hierarchy doc type. In addition to dragging and dropping an application-specific data file into the "Inbox", the system provides two other mechanisms for importing an XML file to be displayed at the server, a Project Translator and an application primary interface (API). The Project Translator is a server application that imports a new project